WaveGAT: Efficient Image Classification using Graph Attention Networks on Wavelet Superpixel Graph

Dissertation submitted to
Visvesvaraya National Institute of Technology, Nagpur
in partial fulfillment of the requirements for the award of the degree

Master of Technology
in
Computer Science & Engineering

by

SURAJ VERMA (MT23MCS018)

under the guidance of

Dr. Gaurav Mishra

Department of Computer Science & Engineering

Visvesvaraya National Institute of Technology

Nagpur 440 010 (India)

JUNE 2025

WaveGAT: Efficient Image Classification using Graph Attention Networks on Wavelet Superpixel Graph

Dissertation submitted to
Visvesvaraya National Institute of Technology, Nagpur
in partial fulfillment of the requirements for the award of
the degree

Master of Technology
in
Computer Science & Engineering

by

SURAJ VERMA (MT23MCS018)

under the guidance of

Dr. Gaurav Mishra

Department of Computer Science & Engineering

Visvesvaraya National Institute of Technology

Nagpur 440 010 (India)

JUNE 2025

©Visvesvaraya National Institute of Technology (VNIT) 2025

Department of Computer Science & Engineering
Visvesvaraya National Institute of Technology, Nagpur

Declaration

I, **Suraj Verma**, hereby declare that this dissertation titled " **WaveGAT: Efficient Image Classification using Graph Attention Networks on Wavelet Superpixel Graph** " is carried out by me in the Department of Computer Science & Engineering of Visvesvaraya National Institute of Technology, Nagpur. The work is original and has not been submitted earlier whole or in part for the award of any degree at this or any other Institution.

Name and signature
Suraj Verma
Date: 13-06-2025

Certificate

This to certify that the dissertation titled " **WaveGAT: Efficient Image Classification using Graph Attention Networks on Wavelet Superpixel Graph** " is submitted by **Suraj Verma** in partial fulfillment of the requirements for the award of the degree of **Master of Technology in Computer Science & Engineering**, VNIT Nagpur. The work is comprehensive, complete, and fit for final evaluation.

Gaurav Mishra
13/06/25.
Dr. Gaurav Mishra

Department of Computer Science
Engineering, VNIT Nagpur

Dr. M. P. Kurhekar

Head, Department of Computer Science
& Engineering, VNIT Nagpur

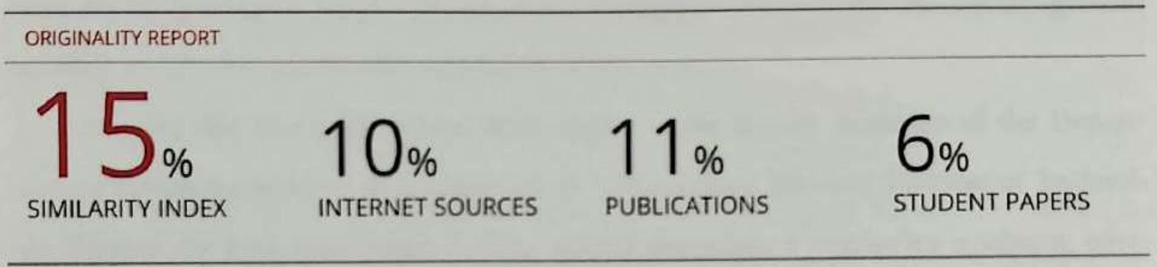
Date:

विभाग प्रमुख
Head
संगणक विज्ञान एवं अभियांत्रिकी विभाग
Dept. of Computer Science & Engg.
वि.रा.प्रौ.सं. नागपुर-१० (भारत)
V.N.I.T., Nagpur-440010. (India)

Department of Computer Science & Engineering
Visvesvaraya National Institute of Technology, Nagpur

Non-Plagiarism Certificate

Certified that M. Tech Dissertation titled “ **WaveGAT: Efficient Image Classification using Graph Attention Networks on Wavelet Superpixel Graph”** submitted by **Suraj Verma**, Enrollment No. **MT23MCS018**, ID No. **30690** has been checked for plagiarism using TURNITIN software and the overlap is found within prescribed limits. A summary of the report is given below.



ORIGINALITY REPORT

15%	10%	11%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

Suraj Verma

Suraj Verma
Enrollment No: MT23MCS018

Gaurav Mishra

Dr. Gaurav Mishra
Assistant Professor
Department of Computer Science & Engineering
VNIT, Nagpur

ACKNOWLEDGEMENT

I would like to express my deepest gratitude and sincere appreciation to all those who have supported and guided me throughout this M.Tech journey.

First and foremost, I am immensely grateful to my supervisor, **Dr. Gaurav Mishra** Sir, for his invaluable guidance, expertise, and encouragement. His constant support, patience, and insightful feedback have been instrumental in shaping this thesis. I am truly fortunate to have the opportunity to work under his mentorship.

I would like to extend my heartfelt thanks to **Dr. M. P. Kurhekar**, Head of Department, for his invaluable support and guidance throughout this journey. His encouragement has been vital in the successful completion of this project.

I would like to extend my heartfelt thanks to the faculty members of the Department of Computer Science & Engineering at Visvesvaraya National Institute of Technology Nagpur for their knowledge-sharing and for providing a conducive academic environment. I am indebted to them for their teachings and guidance which have broadened my perspective and enhanced my understanding of the subject. My sincere appreciation goes to them for their assistance, collaboration, and providing necessary resources for conducting experiments and carrying out this research work. Their contributions have been significant in the successful completion of this thesis.

I would also like to express my gratitude to my family and friends for their unwavering support, understanding, and encouragement throughout this journey. Their love, motivation, and belief in my abilities have been the driving force behind my accomplishments.

Thank you all.

Suraj Verma

ABSTRACT

Image classification, a cornerstone of computer vision, relies heavily on convolutional neural networks, which are well-established for categorizing images into distinct classes. Despite their success, CNNs can be computationally expensive and may not optimally capture the intricate structural information within images. Graph neural networks (GNNs) present a compelling alternative by converting images into graphs, where nodes represent superpixels or image regions and edges represent the spatial relationships between the superpixels, potentially offering a more nuanced understanding of image content. To harness this potential, we explore the use of superpixels—groups of pixels that form perceptually meaningful regions—as a means to reduce data complexity while preserving essential features. Specifically, we utilize WaveMesh, an innovative wavelet-based superpixeling algorithm that generates multiscale superpixels through wavelet decomposition and the extraction of significant coefficients. This method provides a detailed and adaptive representation of image structures, surpassing traditional superpixel techniques such as SLIC. Building on this foundation, we introduce WaveGAT, a novel frameworkher that applies Graph Attention Networks (GAT) to graphs constructed from WaveMesh superpixels. GATs use attention mechanisms to dynamically calculate the level of importance of the neighboring nodes, to help the model to focus on the most important features for classification tasks. Through rigorous experimentation on the FashionMNIST dataset, we compare WaveGAT with other GNN architectures, including SplineCNN, and evaluate the efficacy of different superpixel methods. Our findings demonstrate that WaveGAT achieves exceptional classification performance while maintaining computational efficiency, thereby setting a new benchmark for graph-based image classification methodologies. This thesis contributes to the field by integrating wavelet-based superpixels with attention-based graph networks, offering a promising avenue for future research in efficient and effective image classification.

TABLE OF CONTENTS

Title **Page No.**

ABSTRACT . . . . **iv**

TABLE OF CONTENTS . . . . **v**

LIST OF TABLES . . . . **viii**

LIST OF FIGURES . . . . **ix**

1 INTRODUCTION **1**
- 1.1 Background and Motivation 1
- 1.2 Problem Statement 4
- 1.3 Objectives of the Thesis 4
- 1.4 Key Contributions of the Thesis 5
- 1.5 Thesis Outline 6

2 LITERATURE SURVEY **7**
- 2.1 Image Classification 7
  - 2.1.1 Traditional Methods 8
  - 2.1.2 Deep Learning Approaches 8
- 2.2 Graph Neural Networks 9
  - 2.2.1 Fundamentals of GNNs 10
  - 2.2.2 Graph Attention Networks 10
  - 2.2.3 SplineCNN 11
- 2.3 Superpixel Methods 11

2.3.1 Overview of Superpixels 12

2.3.2 SLIC Algorithm 12

2.3.3 WaveMesh: Wavelet-based Superpixels 13

2.4 Wavelet Transforms in Image Processing 14

2.4.1 Introduction to Wavelets 15

2.4.2 Applications in Image Analysis 15

2.5 Related Work on GNNs for Image Classification 16

3 PROPOSED METHOD 18

3.1 Datasets 18

3.1.1 FashionMNIST 19

3.1.2 Other Datasets 19

3.2 Graph Generation 19

3.2.1 WaveMesh Superpixels 20

3.2.2 SLIC Superpixels 21

3.3 Model Architectures 23

3.3.1 Graph Attention Network (GAT) 24

3.3.2 SplineCNN 25

3.4 Training Procedures 25

3.4.1 Data Preprocessing 26

3.4.2 Hyperparameter Settings 26

3.4.3 Optimization Techniques 26

3.4.4 Evaluation Metrics 27

4 Experiments and Results 28

4.1 Experimental Setup 28

4.1.1 Hardware and Software 28

4.1.2 Dataset Splits 29

4.2 Performance Evaluation 30

4.2.1 Accuracy Metrics 30

4.2.2 Confusion Matrices 31

4.2.3 Per-Class Performance Metrics 32
4.2.4 Computational Efficiency 33
4.3 Comparative Analysis 34
4.3.1 WaveGAT vs. SplineCNN 34
4.3.2 WaveMesh vs. SLIC 35
4.4 Discussion of Results 35

5 CONCLUSION AND FUTURE WORK 37
5.1 Summary of Findings 37
5.2 Limitations 38
5.3 Future Research Directions 39

List of Tables

4.1 Hardware and Software Specifications 29
4.2 Accuracy Metrics for Different Models and Datasets 31
4.3 Per-Class Performance Metrics for WaveMesh + GAT on FashionMNIST 33
4.4 Computational Efficiency Metrics for FashionMNIST 34
4.5 Graph Conversion Metrics 34

List of Figures

2.1 Comparison of SLIC and WaveMesh superpixels on a sample image . . . 14

3.1 Overview of the WaveMesh superpixel generation process 23

4.1 Confusion Matrix for WaveMesh + GAT on FashionMNIST Test Set . . . 32

Chapter 1

INTRODUCTION

1.1 Background and Motivation

The domain of computer vision has witnessed considerable progress in recent times, especially in the area of image classification, which serves as a vital process with numerous uses, including object detection, medical analysis, self-driving technology, and security monitoring. This task requires assigning specific categories to images based on their visual elements, necessitating the recognition and understanding of detailed patterns and characteristics. Such functionality is essential for machines to process and respond to visual information in practical settings, such as spotting items in autonomous vehicles or identifying irregularities in medical scans.

Historically, Convolutional Neural Networks (CNNs) have been the primary tool for image classification, valued for their capacity to autonomously extract layered features from unprocessed pixel data. CNNs function by sweeping convolutional filters over an image, identifying localized patterns and gradually forming more sophisticated representations across multiple layers. While highly effective, CNNs face certain drawbacks. They depend on a rigid, grid-like framework for images, which limits their ability to represent intricate, non-grid relationships among different parts of an image. This structured approach can reduce effectiveness when handling images with unconventional layouts or when long-distance connections within the image are crucial for accurate classification. Moreover, CNNs require substantial computational power for both training and deployment, particularly with extensive datasets or high-definition images. This resource demand creates difficulties in settings with limited capabilities, such as portable devices or time-sensitive systems, where operational efficiency is critical.

To overcome these challenges, alternative strategies have been investigated, with Graph Neural Networks (GNNs) gaining attention as an effective method for handling structured information. GNNs work with graph-based models, where nodes stand for components (e.g., sections of an image) and edges depict connections (e.g., spatial or contextual links), allowing them to account for complex interdependencies. In image classification, images can be converted into graphs, with nodes representing specific areas or superpixels and edges indicating their interactions. This approach enables GNNs to combine localized and broader features, potentially offering improved precision and efficiency over traditional CNN techniques. Recent investigations, including work by Vasudevan et al. [cite the paper], have underscored the value of GNNs in image classification, utilizing graph frameworks to address detailed relationships within images, thus encouraging continued research into this method.

Superpixels are integral to this graph-based approach, as they provide a compact and meaningful representation of images. Superpixels group pixels with similar characteristics, such as color or texture, reducing the complexity of image data while preserving essential structural information. Traditional superpixel methods, such as Simple Linear Iterative Clustering (SLIC), generate superpixels based on color similarity and spatial proximity. However, these methods may struggle to adapt to varying image content and scales, limiting their effectiveness for complex images with intricate details. For instance, SLIC may produce uniform superpixels that fail to capture the hierarchical structure of images with diverse textures or patterns.

To overcome these challenges, wavelet-based superpixel techniques, such as WaveMesh, have been developed. WaveMesh employs wavelet transforms to decompose images into multiple scales, capturing both fine and coarse details. By extracting significant wavelet coefficients and applying quadtree splitting, WaveMesh generates superpixels that adapt to the image's content, providing a more detailed and flexible representation. This multiscale approach is particularly advantageous for images with diverse structures, as it facilitates a hierarchical understanding of visual information, making it well-suited for graph-based classification tasks.

Building on these advancements, this thesis introduces WaveGAT, a novel framework that integrates wavelet-based superpixels with Graph Attention Networks (GATs) for efficient image classification. GATs, a specialized type of GNN, incorporate attention mechanisms that dynamically weigh the importance of neighboring nodes during the learning process, enabling the model to focus on the most relevant features for classification. By applying GATs to graphs constructed from WaveMesh superpixels, WaveGAT leverages the strengths of multiscale image representation and attention-based feature aggregation to enhance classification performance.

The motivation for this research stems from the need for image classification methods that balance accuracy and computational efficiency. In many practical scenarios, such as real-time applications or environments with limited computational resources, the efficiency of the classification model is as critical as its accuracy. By combining wavelet-based superpixels with attention-based GNNs, WaveGAT aims to achieve superior performance while minimizing computational overhead. Furthermore, this work seeks to provide a comprehensive evaluation of WaveGAT across multiple benchmark datasets, including CIFAR10, FashionMNIST, and MNIST, to demonstrate its versatility and robustness. These datasets were selected due to their varying complexities: MNIST consists of simple handwritten digits, FashionMNIST presents more challenging clothing items, and CIFAR10 includes color images of everyday objects, allowing for a thorough assessment of WaveGAT's performance across different levels of image complexity. Through rigorous experimentation and comparison with other GNN architectures, such as SplineCNN, and different superpixel methods, this thesis aims to establish WaveGAT as a state-of-the-art approach for graph-based image classification.

In summary, this research contributes to the ongoing efforts to advance image classification by exploring innovative combinations of superpixel techniques and graph neural networks. By developing and evaluating WaveGAT, we aim to set a new benchmark for efficient and effective image classification, paving the way for future research in this dynamic and impactful field.

1.2 Problem Statement

The primary challenge addressed in this thesis is the development of an efficient and accurate image classification method that leverages the strengths of graph neural networks (GNNs) and advanced superpixel techniques. This challenge encompasses several key aspects: constructing effective graph representations of images that capture both local and global features essential for GNN performance; evaluating and selecting superpixel methods that provide high-quality graph representations while being computationally efficient; designing and optimizing GNN architectures, with a particular focus on Graph Attention Networks (GATs), to achieve superior classification performance on these graph representations; and demonstrating the effectiveness of the proposed framework on multiple benchmark datasets, including CIFAR10, FashionMNIST, and MNIST, through rigorous comparison with existing methods.

1.3 Objectives of the Thesis

The objectives of this thesis are as follows:

1. **Develop the WaveGAT Framework:** To design and implement a novel image classification system that integrates wavelet-based superpixels generated by WaveMesh with Graph Attention Networks (GATs). This framework aims to leverage multiscale image representations and attention mechanisms to enhance classification accuracy while maintaining computational efficiency, addressing the limitations of traditional convolutional neural networks (CNNs) in capturing complex, non-Euclidean relationships within images.

2. **Evaluate Performance on Benchmark Datasets:** To assess the classification performance and computational efficiency of WaveGAT across three benchmark datasets: CIFAR10, FashionMNIST, and MNIST. This evaluation will provide a comprehensive understanding of the framework's robustness and versatility across diverse image types, ranging from simple grayscale digits to complex color images, thereby demonstrating its applicability in various real-world scenarios.

3. **Compare with Other GNN Architectures:** To benchmark WaveGAT against other graph neural network architectures, particularly SplineCNN, using consistent datasets and superpixel methods. This comparison will elucidate the relative strengths and weaknesses of WaveGAT, highlighting its potential advantages in terms of accuracy and efficiency for graph-based image classification tasks.

4. **Analyze the Impact of Superpixel Methods:** To investigate the influence of different superpixel methods, specifically WaveMesh and Simple Linear Iterative Clustering (SLIC), on the quality of graph representations and subsequent classification performance. This analysis will include evaluating computational costs and the suitability of each method for different image complexities, providing insights into optimal superpixel selection for GNN-based classification.

5. **Provide Insights into Accuracy-Efficiency Trade-offs:** To explore the balance between classification accuracy and computational resources required by graph-based methods compared to traditional CNNs. This objective aims to identify scenarios where WaveGAT offers significant advantages, particularly in resource-constrained environments, and to provide guidelines for deploying graph-based classification systems in practical applications.

By achieving these objectives, this thesis aims to advance the field of image classification by demonstrating the potential of graph-based methods, particularly through the integration of wavelet-based superpixels and attention-based graph neural networks, thereby providing efficient and effective solutions for various applications.

1.4 Key Contributions of the Thesis

The key contributions of this research include:

- **Development of WaveGAT Framework**: Creation of a novel image classification framework that combines wavelet-based superpixels with Graph Attention Networks, leveraging multiscale image representations and attention mechanisms for enhanced performance.

- **Evaluation Across Multiple Datasets**: Comprehensive assessment of WaveGAT on CIFAR10, FashionMNIST, and MNIST datasets, demonstrating its effectiveness across diverse image types and complexities.

- **Comparative Analysis with SplineCNN**: Detailed comparison with SplineCNN and other GNN architectures, highlighting WaveGAT's advantages in terms of accuracy and efficiency.

- **Assessment of Superpixel Methods**: In-depth evaluation of WaveMesh and SLIC superpixels, providing insights into their suitability for graph-based classification tasks.

- **Analysis of Computational Efficiency**: Examination of training and inference times to evaluate the practicality of the proposed methods in real-world scenarios.

1.5 Thesis Outline

This thesis is organized as follows:

- **Chapter 2: Literature Survey** – Reviews existing work on image classification, graph neural networks, superpixel methods, and their applications in computer vision.

- **Chapter 3: Methodology** – Details the development of the WaveGAT framework, including the generation of graphs using WaveMesh and SLIC superpixels, and the architecture of the GAT model.

- **Chapter 4: Experiments and Results** – Presents the experimental setup, results on the CIFAR10, FashionMNIST, and MNIST datasets, and comparative analysis with other methods.

- **Chapter 5: Conclusion and Future Work** – Summarizes the findings, discusses limitations, and suggests directions for future research.

Chapter 2

LITERATURE SURVEY

In this chapter, we review the existing literature related to our work on graph-based image classification using the WaveGAT framework. Section 2.1 explores image classification, covering both traditional methods and deep learning approaches. Section 2.2 examines the literature on graph neural networks, providing the fundamentals and detailing specific architectures like Graph Attention Networks and SplineCNN. Section 2.3 investigates superpixel methods, including an overview, the SLIC algorithm, and the WaveMesh approach that leverages wavelet transforms. Section 2.4 examines wavelet transforms in image processing, introducing wavelet theory and their applications in image analysis. Finally, Section 2.5 reviews related work on graph neural networks for image classification, highlighting how our research builds on and extends these efforts.

2.1 Image Classification

Image classification entails assigning specific categories to images based on their visual elements, serving as a vital component of computer vision. This process underpins numerous applications, such as identifying objects, analyzing medical images, enabling autonomous vehicles, and supporting surveillance technologies. Over the years, the discipline has undergone substantial changes, moving from early techniques that depended on manually extracted features to cutting-edge deep learning methods. This section examines both traditional and modern strategies for classifying images, exploring their advantages and drawbacks to provide context for the emergence of graph-oriented techniques.

2.1.1 Traditional Methods

Prior to the development of deep learning, image classification leaned heavily on manually crafted feature extraction paired with conventional machine learning algorithms. Initial efforts concentrated on obtaining durable and distinguishing traits from images to reflect their visual essence. Key methods include:

- **Histogram of Oriented Gradients (HOG)**: Developed by [7], HOG evaluates the arrangement of gradient directions within localized sections of an image, proving especially useful for recognizing shapes and objects, such as in pedestrian identification tasks.
- **Scale-Invariant Feature Transform (SIFT)**: Introduced by [21], SIFT pinpoints features that remain stable across changes in scale and orientation, delivering dependable attributes for image comparison and identification.
- **Speeded Up Robust Features (SURF)**: Proposed by [3], SURF provides a swifter option compared to SIFT by employing integral images, retaining resilience while boosting processing speed.

These characteristics were generally processed by classifiers like Support Vector Machines (SVMs) [6], Random Forests [4], or k-Nearest Neighbors (k-NN) [2]. Moreover, the Bag of Words (BoW) framework [25] portrayed images as distributions of visual terms, allowing categorization based on the prevalence of features. Though these approaches excelled in certain contexts, they necessitated considerable manual effort in designing features and often failed to adapt to varied datasets due to their dependence on fixed attributes, restricting their utility in intricate, practical settings.

2.1.2 Deep Learning Approaches

The arrival of deep learning, particularly through convolutional neural networks (CNNs), ushered in a revolutionary change in image classification. CNNs independently extract layered features from unprocessed pixel information, removing the need for handcrafted feature design. Notable developments in CNN evolution include:

- **LeNet-5**: Created by [19], LeNet-5 was among the pioneering CNNs, tailored for recognizing handwritten numbers in the MNIST dataset, establishing a basis for subsequent designs.
- **AlexNet**: Presented by [18], AlexNet delivered exceptional results in the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), elevating deep learning's prominence with its extensive structure and innovations like dropout and ReLU activation.
- **VGGNet**: Formulated by [24], VGGNet advanced the field with deeper configurations using smaller convolutional filters, enhancing precision on expansive datasets like ImageNet.
- **ResNet**: Devised by [14], ResNet incorporated residual links to facilitate the training of exceptionally deep networks, securing top performance by overcoming gradient diminishment challenges.

CNNs have emerged as the leading method for image classification, owing to their capacity to discern detailed patterns and manage extensive data efficiently. Nonetheless, they presuppose a uniform, grid-like arrangement for images, which hampers their ability to depict intricate, non-grid-based connections between image segments. Additionally, CNNs require significant computational resources for training and operation, particularly with high-definition images or in environments with limited capabilities. These constraints have driven researchers to investigate alternative solutions, such as graph neural networks, which excel at representing structural relationships within images.

2.2 Graph Neural Networks

Graph Neural Networks (GNNs) have emerged as a powerful framework for modeling structured data, offering a flexible approach to capturing relationships in non-Euclidean spaces. In the context of image classification, GNNs represent images as graphs, with nodes corresponding to image regions or superpixels and edges encoding spatial or semantic relationships. This section reviews the fundamentals of GNNs and two specific architectures relevant to this thesis: Graph Attention Networks (GATs) and SplineCNN.

2.2.1 Fundamentals of GNNs

A graph $G = (V, E)$ consists of a set of nodes V and edges E, where each node is associated with a feature vector, and edges represent relationships between nodes. GNNs operate by propagating information across the graph through a message-passing framework, where each node aggregates features from its neighbors to update its own representation. This process is iterated over multiple layers to capture higher-order relationships. Key GNN architectures include:

- **Graph Convolutional Networks (GCNs)**: Introduced by [15], GCNs apply convolution operations on graphs by aggregating normalized features from neighboring nodes, suitable for tasks like node classification.
- **GraphSAGE**: Proposed by [13], GraphSAGE uses sampling to aggregate features from a fixed number of neighbors, enabling scalability to large graphs.
- **Graph Isomorphism Network (GIN)**: Developed by [31], GIN is designed to be as expressive as the Weisfeiler-Lehman test for graph isomorphism, improving performance on graph classification tasks.

GNNs are particularly advantageous for image classification when images are represented as graphs, as they can model complex dependencies that CNNs may overlook. However, the effectiveness of GNNs depends on the quality of the graph representation and the choice of aggregation mechanisms.

2.2.2 Graph Attention Networks

Graph Attention Networks (GATs), introduced by [28], enhance GNNs by incorporating attention mechanisms that dynamically weigh the importance of neighboring nodes. In GATs, attention coefficients are computed using a shared attention mechanism, typically involving a linear transformation, a non-linear activation (e.g., LeakyReLU), and a softmax function to normalize weights. This allows the model to focus on the most relevant neighbors for each node, improving performance on tasks where node importance varies.

GATs have been successfully applied to various tasks, including node classification and graph classification, due to their ability to adaptively aggregate features. In the context of image classification, GATs can be applied to graphs constructed from image regions or superpixels, enabling the model to prioritize visually significant regions. The attention mechanism makes GATs particularly suitable for the WaveGAT framework proposed in this thesis, as it enhances the model's ability to focus on critical image features.

2.2.3 SplineCNN

SplineCNN, proposed by [11], is a GNN architecture that uses continuous B-spline kernels to parameterize convolution operations on graphs. Unlike traditional GNNs that rely on discrete aggregations, SplineCNN employs smooth, continuous kernels defined over the graph's edge attributes, allowing for fine-grained feature extraction. This approach is particularly effective for tasks requiring precise modeling of spatial relationships, such as image classification with superpixel-based graphs.

In the context of this thesis, SplineCNN serves as a baseline for comparison with the WaveGAT framework, as it has been used with superpixel graphs in prior work [27]. The continuous nature of B-spline kernels makes SplineCNN a robust choice for graph-based image classification, though it may lack the adaptive weighting provided by attention mechanisms in GATs.

2.3 Superpixel Methods

Superpixels are groups of pixels that share similar characteristics, such as color or texture, and serve as an intermediate representation between pixels and objects. They reduce the complexity of image data while preserving essential structural information, making them valuable for tasks like segmentation, object detection, and image classification. This section reviews the concept of superpixels and two specific methods relevant to this thesis: SLIC and WaveMesh.

2.3.1 Overview of Superpixels

Superpixels provide a compact representation of images by grouping pixels into perceptually meaningful regions, reducing the number of elements to process and facilitating efficient computation. They are widely used in computer vision to improve the performance of algorithms by focusing on regions rather than individual pixels. Common superpixel methods include:

- **Normalized Cuts**: Proposed by [23], this method uses spectral clustering to partition images based on global image properties.
- **Felzenszwalb's Method**: Developed by [10], this graph-based approach uses minimum spanning trees to generate superpixels that adhere to image boundaries.
- **SLIC (Simple Linear Iterative Clustering)**: Introduced by [1], SLIC is a popular method due to its simplicity and efficiency, as discussed below.

Superpixels are particularly relevant for graph-based image classification, as they can serve as nodes in a graph, with edges representing spatial or semantic relationships. The quality of superpixels significantly impacts the performance of subsequent processing steps, necessitating methods that balance accuracy and computational efficiency.

2.3.2 SLIC Algorithm

The Simple Linear Iterative Clustering (SLIC) algorithm, proposed by [1], generates superpixels by clustering pixels in a five-dimensional space that combines color (in LAB color space) and spatial coordinates (XY). SLIC uses k-means clustering with a distance measure that balances color similarity and spatial proximity, controlled by a compactness parameter. The algorithm initializes cluster centers on a regular grid and iteratively refines them to produce superpixels that adhere to image boundaries.

The SLIC algorithm can be summarized as follows:

algorithm algpseudocode amsmath

SLIC is widely adopted due to its simplicity, speed, and ability to generate regular, compact superpixels. However, it may struggle with images containing complex textures

Algorithm 1 SLIC Superpixel Segmentation

```
1: **Input**: Image $I$, number of superpixels $k$, compactness $m$
2: **Output**: Superpixel labels for each pixel in $I$
3: Convert $I$ to LAB color space
4: Initialize cluster centers $C_i = [l_i, a_i, b_i, x_i, y_i]$ on grid with step $S = \sqrt{N/k}$, $N$ = number of pixels
5: Move centers to lowest gradient in $3 \times 3$ neighborhood
6: Set label $l_i = -1$, distance $d_i = \infty$ for each pixel $i$
7: **for** each center $C_i$ **do**
8:     **for** each pixel $p$ in $2S \times 2S$ region around $C_i$ **do**
9:         $d_c = \sqrt{(l_p - l_i)^2 + (a_p - a_i)^2 + (b_p - b_i)^2},\ d_s = \sqrt{(x_p - x_i)^2 + (y_p - y_i)^2}$
10:        $D = d_c + (m/S) \cdot d_s$
11:        **if** $D < d_p$ **then**
12:            $d_p \leftarrow D$, $l_p \leftarrow i$
13:        **end if**
14:    **end for**
15: **end for**
16: Update centers with mean $[l, a, b, x, y]$ of assigned pixels
17: Repeat steps 7–11 until convergence or for 10 iterations
18: Enforce connectivity by merging small regions to nearest superpixel
19: **Return** superpixel labels
```

or varying scales, as it relies on uniform clustering criteria. In this thesis, SLIC serves as a baseline superpixel method for comparison with WaveMesh, implemented to generate graphs for GNN-based classification.

2.3.3 WaveMesh: Wavelet-based Superpixels

WaveMesh, introduced by [27], is a novel superpixel method that leverages wavelet transforms to generate multiscale superpixels. Unlike traditional methods, WaveMesh decomposes images into multiple scales using wavelet transforms (e.g., Haar wavelets), identifying significant coefficients that represent important image features. These coefficients guide a quadtree splitting process to create superpixels that adapt to the image's content, capturing both fine and coarse details.

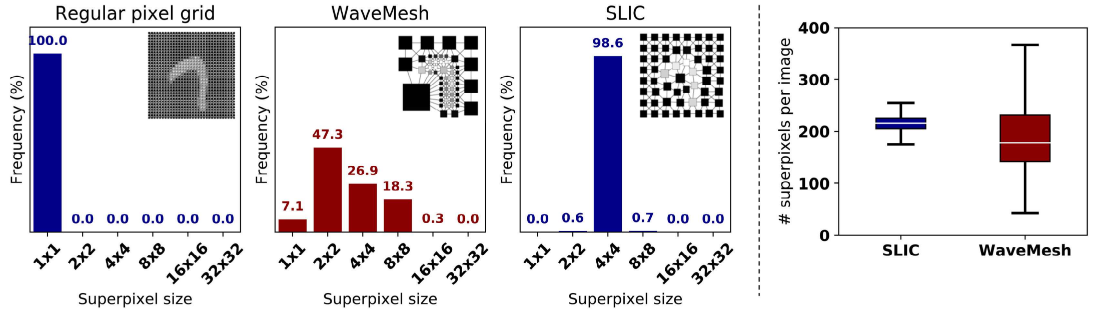
The WaveMesh algorithm can be summarized as follows:

To illustrate the differences between SLIC and WaveMesh superpixels, Figure 2.1 shows a comparison from Vasudevan et al. (2023), highlighting WaveMesh's ability to capture multiscale details compared to SLIC's uniform clustering.

This multiscale approach makes WaveMesh particularly suitable for images with in-

Algorithm 2 WaveMesh Algorithm

```
procedure WAVEMESH($I$, $N$)
    Decompose image $I$ into wavelet representation using Haar wavelets up to scale $S = \log_2 N$
    Estimate noise variance $\sigma^2_{I_\le}$ iteratively
    Compute threshold $T = \sqrt{2\sigma^2_{I_\le} \ln N^2}$
    Threshold wavelet coefficients: $\tilde{I}^{(s,d)}_{>}(\boldsymbol{x}_s) = \tilde{I}^{(s,d)}(\boldsymbol{x}_s)$ if $|\tilde{I}^{(s,d)}(\boldsymbol{x}_s)| \ge T$, else 0
    Construct quadtree from wavelet coefficients, tagging nodes based on filter mask
    Generate superpixels by recursively splitting tagged regions into $2 \times 2$ superpixels
    Build region adjacency graph (RAG) from superpixels
end procedure
```



(a) (b)

Figure 2.1: Comparison of SLIC and WaveMesh superpixels on a sample image [27].

tricate structures, as it provides a hierarchical representation that enhances graph-based processing. In this thesis, WaveMesh is used to construct graphs for the WaveGAT framework, enabling detailed and adaptive image representations.

2.4 Wavelet Transforms in Image Processing

Wavelet transforms are mathematical tools that decompose signals into components at different scales, providing both frequency and spatial information. They are widely used in image processing due to their ability to capture multiscale features, making them relevant for superpixel generation and image analysis tasks.

2.4.1 Introduction to Wavelets

Wavelets are functions that satisfy specific mathematical criteria, enabling the representation of data at multiple resolutions. Unlike Fourier transforms, which provide frequency information without spatial localization, wavelets offer both frequency and spatial localization, making them ideal for analyzing non-stationary signals like images. A wavelet transform decomposes an image into approximation (low-frequency) and detail (high-frequency) coefficients across multiple scales.

Common wavelet families include:

- **Haar Wavelets**: The simplest wavelets, using step functions for decomposition [12].
- **Daubechies Wavelets**: Orthogonal wavelets with compact support, offering smoother representations [8].
- **Mexican Hat Wavelets**: Used in continuous wavelet transforms for feature detection [22].

In image processing, 2D wavelet transforms are applied to decompose images into a hierarchy of subbands, enabling multiscale analysis. The Haar wavelet, used in WaveMesh, is particularly effective due to its simplicity and computational efficiency.

2.4.2 Applications in Image Analysis

Wavelet transforms have numerous applications in image processing, including:

- **Denoising**: Thresholding wavelet coefficients to remove noise while preserving image details [9].
- **Compression**: JPEG2000 uses wavelet transforms for efficient image compression, achieving high quality at low bit rates [26].
- **Feature Extraction**: Wavelet coefficients serve as features for tasks like texture analysis and object detection [22].
- **Superpixel Generation**: Wavelets guide the creation of adaptive superpixels, as in WaveMesh, by identifying significant image regions [27].

In this thesis, wavelet transforms are central to the WaveMesh superpixel method, enabling the construction of multiscale graphs for GNN-based image classification.

2.5 Related Work on GNNs for Image Classification

The application of graph neural networks to image classification has gained significant attention in recent years, driven by their ability to model complex relationships in non-Euclidean spaces. This section reviews key studies that have explored GNNs for image classification, particularly those using superpixels or similar graph construction methods, to contextualize the contributions of the WaveGAT framework.

A seminal work by [27] proposed using multiscale wavelet superpixels (WaveMesh) to construct graphs for image classification with GNNs. The authors compared SplineCNN with different pooling strategies (no pooling, GraclusPool, WavePool) on datasets like CIFAR10, FashionMNIST, and MNIST, finding that WavePool outperformed traditional pooling methods. This work directly inspires the WaveGAT framework, which extends the use of WaveMesh superpixels to Graph Attention Networks, leveraging attention mechanisms for improved performance.

Other studies have explored alternative graph construction methods for image classification. For instance, [16] used pixel-based graphs, where each pixel is a node, and applied GCNs to classify images, demonstrating the potential of graph-based methods but highlighting computational challenges due to large graph sizes. [32] proposed a dynamic graph construction approach that learns graph structures during training, achieving competitive performance on ImageNet but requiring significant computational resources.

Attention-based GNNs have also been applied to image classification. [29] used GATs to classify images by constructing graphs from image patches, showing improved performance over CNNs on datasets with complex structures. Similarly, [20] combined superpixels with GATs for semantic segmentation, a task closely related to classification, demonstrating the effectiveness of attention mechanisms in focusing on relevant image regions.

Research on superpixel-based GNNs includes [5], who used SLIC superpixels to construct graphs for GCN-based classification, achieving efficiency but limited by SLIC's uniform clustering. In contrast, wavelet-based approaches, like WaveMesh, offer adaptive superpixels that better capture image details, as explored in this thesis.

These studies highlight the potential of GNNs for image classification but also underscore challenges, such as the computational cost of graph construction and the need for effective superpixel methods. The WaveGAT framework builds on these works by integrating wavelet-based superpixels with attention-based GNNs, aiming to achieve a balance of accuracy and efficiency across diverse datasets.

Chapter 3

PROPOSED METHOD

This chapter delineates the methodology employed to develop and evaluate the WaveGAT framework, which integrates wavelet-based superpixels with Graph Attention Networks (GATs) for efficient image classification. The methodology encompasses the selection and preparation of datasets, the generation of graph representations using WaveMesh and SLIC superpixel methods, the design of the WaveGAT and SplineCNN model architectures, and the training procedures used to optimize and assess performance. Section 3.1 describes the datasets, detailing their characteristics and relevance to the classification task. Section 3.2 outlines the graph generation process, including wavelet decomposition, superpixel extraction, and graph construction for both WaveMesh and SLIC methods. Section 3.3 details the architectures of the GAT and SplineCNN models, focusing on their key components and operational mechanisms. Section 3.4 explains the training procedures, covering data preprocessing, hyperparameter settings, optimization techniques, and evaluation metrics. The methodology is designed to ensure reproducibility and robustness, aligning with the objectives of evaluating graph-based image classification techniques.

3.1 Datasets

This section describes the datasets utilized to evaluate the WaveGAT framework, focusing on their characteristics, class distributions, and relevance to the image classification task.

3.1.1 FashionMNIST

The FashionMNIST dataset, a widely recognized benchmark for image classification, comprises 70,000 grayscale images (1 channel) of size 28×28 pixels, divided into 60,000 training images and 10,000 test images. It includes 10 classes of clothing items: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot, with 6,000 images per class in the training set and 1,000 per class in the test set. FashionMNIST was selected due to its moderate complexity, which challenges models to distinguish fine-grained visual differences among clothing items, making it an ideal testbed for evaluating the WaveGAT framework's ability to capture subtle patterns in grayscale images [30].

3.1.2 Other Datasets

To assess the versatility of the WaveGAT framework, we also utilized the CIFAR10 and MNIST datasets. CIFAR10 consists of 60,000 color images (3 channels, RGB) of size 32×32 pixels, with 50,000 training images and 10,000 test images across 10 classes: Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, and Truck, with 5,000 images per class in the training set and 1,000 per class in the test set. This dataset tests the framework's performance on color images with diverse textures and backgrounds [17]. MNIST contains 70,000 grayscale images (1 channel) of size 28×28 pixels, representing handwritten digits (0 through 9), with 60,000 training images and 10,000 test images, approximately 6,000 images per class in the training set and 1,000 per class in the test set. MNIST serves as a simpler baseline to evaluate the framework's performance on well-structured, low-complexity images [19]. These datasets were chosen to provide a comprehensive evaluation across varying image complexities and channel types, ensuring robust assessment of the WaveGAT framework.

3.2 Graph Generation

Graph generation is a pivotal component of the WaveGAT framework, transforming images into graph structures suitable for processing by graph neural networks. We employed two superpixel methods—WaveMesh and SLIC—to generate these graphs, each with distinct approaches to decomposition, extraction, and construction.

3.2.1 WaveMesh Superpixels

WaveMesh, introduced by [27], is a novel superpixel method that leverages wavelet transforms to generate multiscale superpixels, capturing both fine and coarse image details. The process involves three key steps: wavelet decomposition, superpixel extraction, and graph construction.

Wavelet Decomposition

The WaveMesh method begins with a 2D discrete wavelet transform (DWT) using Haar wavelets, as specified in the configuration (`wname='haar'`). For an image I, the DWT decomposes the image into approximation (cA_j) and detail coefficients (cH_j, cV_j, cD_j) at multiple levels j, capturing low-frequency (coarse) and high-frequency (fine) details, respectively. The decomposition is performed up to a maximum level determined by the image size and a safety margin (`max_level_safety_margin=1`), with images padded to power-of-2 dimensions (`power2_pad=True`) to ensure compatibility. Stabilization noise (`stabilization_noise=`1.0×10^{-2}) is added to detail coefficients to mitigate numerical issues. Mathematically, the DWT at level j is expressed as:

$$[cA_j, cH_j, cV_j, cD_j] = \text{DWT}(I),$$

where cA_j represents the approximation coefficients, and cH_j, cV_j, and cD_j capture horizontal, vertical, and diagonal details, respectively.

Superpixel Extraction

Superpixels are extracted by identifying significant wavelet coefficients through iterative thresholding based on the Median Absolute Deviation (MAD), adjusted by a threshold multiplier (`thresh_mult=0.3`). The threshold T is computed as:

$$T = \text{MAD}(cH_1, cV_1, cD_1) \times 0.3,$$

where cH_1, cV_1, and cD_1 are the detail coefficients at the finest level. Coefficients satisfying $|cH_j| \geq T$, $|cV_j| \geq T$, or $|cD_j| \geq T$ are tagged as significant, and their ancestors are propagated up the quadtree to the coarsest level. A quadtree splitting process then generates superpixels by recursively dividing regions containing significant coefficients until a minimum size (`min_size=2`) is reached, ensuring adaptive segmentation that captures multiscale image features. Algorithm 2 outlines the core steps of this process.

Algorithm 3 WaveMesh Superpixel Segmentation

```
Input: Image $I$, wavelet name (e.g., Haar), threshold multiplier $m$, minimum size $s$
Output: Superpixel labels for each pixel in $I$
Perform discrete wavelet transform (DWT) on preprocessed $I$ up to level $l_{\max}$
for each channel in $I$ do
    Obtain coefficients $cA$, $(cH, cV, cD)$ at each level, adding stabilization noise to details
end for
Compute threshold $T$ using MAD on finest-level detail coefficients, adjusted by $m$
Tag significant coefficients ($|\text{cH}| \geq T$, $|\text{cV}| \geq T$, or $|\text{cD}| \geq T$) across levels
Propagate tags to ancestors up to the coarsest level using quadtree traversal
Initialize superpixel label array $L$, set root size $S \leftarrow 2^{l_{\max}-1}$, label counter label $\leftarrow 1$
for each position $(x, y)$ in $L$ at intervals of $S$ do
    Recursively split regions with tagged coefficients into quadrants until size $< s$ or level $< 1$, assigning labels
end for
Return Superpixel labels for each pixel in $I$
```

Graph Construction

Each superpixel is represented as a node in the graph, with features including mean RGB values, standard deviation, area, and centroid positions, totaling 6 features per node. Edges connect adjacent superpixels, with attributes such as relative centroid distances, forming a graph $G = (V, E)$, where V is the set of superpixel nodes and E represents spatial relationships. This graph structure enables the capture of complex image patterns for processing by graph neural networks, as described in [27].

3.2.2 SLIC Superpixels

As a baseline, the Simple Linear Iterative Clustering (SLIC) algorithm is used to generate uniform superpixels, providing a computationally efficient alternative to WaveMesh

[1].

SLIC Algorithm

The SLIC algorithm clusters pixels in a five-dimensional space combining LAB color values and spatial coordinates (XY). It initializes cluster centers on a regular grid with step size $S = \sqrt{N/k}$, where N is the number of pixels and k is the desired number of superpixels. These centers are perturbed to low-gradient positions to avoid edges, and pixels are iteratively assigned to the nearest center based on a distance measure that balances color similarity and spatial proximity, controlled by a compactness parameter. The process is outlined in [1].

Algorithm 4 SLIC Superpixel Segmentation

```
1:  **Input**: Image $I$, number of superpixels $k$, compactness $m$
2:  **Output**: Superpixel labels for each pixel in $I$
3:  Convert image $I$ to LAB color space
4:  Initialize cluster centers $C_i = [l_i, a_i, b_i, x_i, y_i]$ on a regular grid with step size $S = \sqrt{N/k}$, where $N$ is the number of pixels in $I$
5:  Move cluster centers to the lowest gradient position in a 3x3 neighborhood
6:  Initialize label array $l_i = -1$ for each pixel $i$
7:  Initialize distance array $d_i = \infty$ for each pixel $i$
8:  for each cluster center $C_i$ do
9:      for each pixel $p$ in a $2S \times 2S$ region around $C_i$ do
10:         Compute color distance $d_c = \sqrt{(l_p - l_i)^2 + (a_p - a_i)^2 + (b_p - b_i)^2}$
11:         Compute spatial distance $d_s = \sqrt{(x_p - x_i)^2 + (y_p - y_i)^2}$
12:         Compute total distance $D = d_c + \frac{m}{S} d_s$
13:         if $D < d_p$ then
14:             Set $d_p \leftarrow D$
15:             Set $l_p \leftarrow i$
16:         end if
17:     end for
18: end for
19: Update cluster centers by computing the mean $[l, a, b, x, y]$ vector of all pixels assigned to each cluster
20: Repeat Steps 8–16 until convergence or for a fixed number of iterations (typically 10)
21: Enforce connectivity by merging small or disconnected regions to the nearest superpixel
22: **Return** Superpixel labels for each pixel in $I$
```

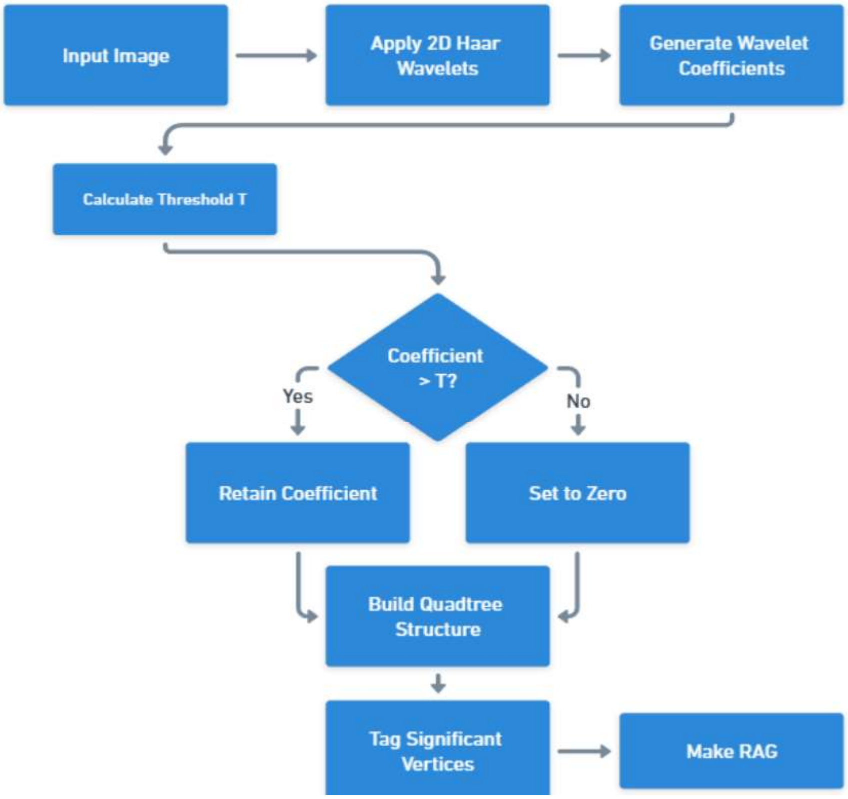

Input Image
Apply 2D Haar Wavelets
Generate Wavelet Coefficients
Calculate Threshold T
Coefficient > T?
Yes
No
Retain Coefficient
Set to Zero
Build Quadtree Structure
Tag Significant Vertices
Make RAG

Figure 3.1: Overview of the WaveMesh superpixel generation process

Graph Construction

SLIC superpixels are used to construct a graph similar to WaveMesh, where each superpixel forms a node with features such as mean RGB values, standard deviation, area, and centroid positions, totaling 6 features per node. Edges connect adjacent superpixels based on spatial adjacency, with attributes like relative centroid distances, forming a graph $G = (V, E)$. This graph is less computationally intensive to generate than WaveMesh but may capture fewer multiscale details, as discussed in Chapter 2.

3.3 Model Architectures

This section describes the architectures of the two graph neural network models evaluated: Graph Attention Network (GAT) for the WaveGAT framework and SplineCNN as a baseline.

3.3.1 Graph Attention Network (GAT)

The WaveGAT framework leverages GATs to exploit attention mechanisms for dynamic feature aggregation in graph-based image classification [28].

Architecture Details

The GAT model consists of multiple graph attention layers, each employing multi-head attention to capture diverse relationships among nodes. The input graph, derived from WaveMesh or SLIC superpixels, has nodes with 6 features: mean RGB values (3 features), standard deviation, area, and centroid positions (x, y coordinates). Each GAT layer applies attention-based aggregation, followed by batch normalization to stabilize training. Specifically, the architecture includes 4 layers (`num_layers=4`) with a hidden dimension of 256 (`hidden_dim=256`), multi-head attention with 4 heads in the first three layers and 1 head in the last layer, and a dropout rate of 0.5 (`dropout=0.5`) to prevent overfitting. The output of the final GAT layer is processed through a global attention pooling layer to produce a single graph-level representation, which is fed into a classification head comprising two linear layers (from hidden dimension to hidden dimension, then to 10 classes), ReLU activation, and dropout (fixed at 0.5 in the classification head). The architecture outputs probabilities for 10 classes, corresponding to the dataset classes (e.g., FashionMNIST's 10 clothing categories).

Attention Mechanism

The attention mechanism in GAT computes coefficients to weigh the importance of neighboring nodes. For nodes i and j, the attention coefficient α_{ij} is calculated as:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i||\mathbf{W}\mathbf{h}_j]))}{\sum_{k\in\mathcal{N}_i}\exp(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i||\mathbf{W}\mathbf{h}_k]))},$$

where $\mathbf{h}_i$ is the feature vector of node i, $\mathbf{W}$ is a learnable weight matrix, $\mathbf{a}$ is a learnable attention vector, $||$ denotes concatenation, and $\mathcal{N}_i$ is the set of neighbors of node i. This mechanism allows the model to focus on relevant neighbors, enhancing classification performance by dynamically weighting node contributions based on their contextual importance.

3.3.2 SplineCNN

SplineCNN serves as a baseline, utilizing B-spline kernels for graph convolution operations [11].

Architecture Details

The SplineCNN model comprises three SplineConv layers with 32, 64, and 128 features, respectively, using a kernel size of [3,3]. The input graph has nodes with 3 features (mean RGB values) and edges with normalized spatial attributes (relative centroid distances, normalized to [0, 1]). Each SplineConv layer applies convolution operations using B-spline kernels, followed by ReLU activation to introduce non-linearity.

B-Spline Kernels

B-spline kernels parameterize convolution operations as smooth, continuous functions over graph edges, defined by learnable basis functions. For a graph with edge attributes e_{ij}, the convolution at node i aggregates features from neighbors $j \in \mathcal{N}_i$:

$$\mathbf{h}'_i = \sum_{j \in \mathcal{N}_i} \sum_{k} w_k B_k(e_{ij}) \mathbf{h}_j,$$

where B_k are B-spline basis functions, w_k are learnable weights, and $\mathbf{h}_j$ are the feature vectors of neighboring nodes. This approach ensures robust feature extraction by modeling continuous relationships across edges, contrasting with GAT's attention-based aggregation, and provides a strong baseline for comparison.

3.4 Training Procedures

This section outlines the training procedures for the WaveGAT and SplineCNN models, covering data preprocessing, hyperparameter settings, optimization techniques, and evaluation metrics.

3.4.1 Data Preprocessing

Images were preprocessed to ensure consistency across datasets. Pixel intensities were normalized to the range [0, 1] to standardize input values. For WaveMesh processing, images were converted to RGB format (for consistency across datasets, even for grayscale images like FashionMNIST and MNIST) and padded to 32×32 pixels to ensure compatibility with wavelet decomposition, as specified in the configuration (`power2_pad=True`). Graphs were precomputed and stored in the output directory (`output_dir=./dataset/graph_c` loaded in batches of 32 (`batch_size=32`) using 4 worker processes (`num_workers=4`) for efficient training. The training data was further split into training and validation sets with an 80/20 ratio, ensuring robust evaluation during training.

3.4.2 Hyperparameter Settings

Hyperparameters for the WaveGAT model were optimized using Optuna, a hyperparameter optimization framework, to determine the optimal settings for hidden dimensions (`hidden_dim=256`), number of layers (`num_layers=4`), number of attention heads, learning rate (`lr=0.0005`), weight decay (`weight_decay=4e-5`), dropout rate (`dropout=0.5`), batch size (`batch_size=64`), and early stopping patience (`patience=30`). For SplineCNN, standard hyperparameters were used: a learning rate of 0.01, batch size of 64, and early stopping with a patience of 10 epochs, as specified in the model description. These settings were tuned to maximize validation accuracy, ensuring a fair comparison between models, as detailed in Chapter 4.

3.4.3 Optimization Techniques

The WaveGAT model was trained using the AdamW optimizer with weight decay to prevent overfitting, coupled with a ReduceLROnPlateau scheduler to adjust the learning rate based on validation loss, reducing it by a factor of 0.5 if no improvement was observed for 10 epochs, with a minimum learning rate of 1×10^{-6}. Early stopping with a patience of 30 epochs was employed to halt training when performance plateaued. SplineCNN used the Adam optimizer with a fixed learning rate of 0.01 and a MultiStepLR scheduler, reducing the learning rate by a factor of 0.1 at epochs 15 and 25, with early stopping after 10 epochs of no improvement. Both models utilized CrossEntropyLoss with label smoothing (0.1) to enhance generalization by reducing overconfidence in predictions.

3.4.4 Evaluation Metrics

Model performance was evaluated using a comprehensive set of metrics to assess both effectiveness and practicality. Classification performance was measured using accuracy, precision, recall, and F1-score on the test set, providing a detailed understanding of per-class performance. Confusion matrices were generated to analyze classification errors, particularly for visually similar classes like Shirt and T-shirt/top in FashionMNIST, as discussed in Chapter 4. Computational efficiency was assessed through training time per epoch, inference time per graph, and graph conversion time, ensuring a balanced evaluation of model efficiency alongside accuracy. These metrics collectively provide a robust assessment of the WaveGAT framework's performance across diverse datasets and scenarios.

Chapter 4

Experiments and Results

In this chapter, we present the experimental evaluation of the WaveGAT framework, which integrates wavelet-based superpixels with Graph Attention Networks (GATs) for efficient image classification. The experiments were designed to assess the performance of WaveGAT against a baseline SplineCNN model, using both WaveMesh and SLIC superpixel methods across three benchmark datasets: FashionMNIST, CIFAR10, and MNIST. Section 4.1 details the experimental setup, including hardware, software, and dataset splits. Section 4.2 presents the performance evaluation, covering accuracy metrics, confusion matrices, per-class performance metrics, and computational efficiency. Section 4.3 provides a comparative analysis of WaveGAT versus SplineCNN and WaveMesh versus SLIC. Finally, Section 4.4 discusses the results, highlighting key findings and their implications.

4.1 Experimental Setup

This section outlines the hardware and software environment used for the experiments, as well as the dataset splits employed to ensure robust evaluation.

4.1.1 Hardware and Software

The experiments were conducted on a high-performance computing system with specifications detailed in Table 4.1. This setup ensured efficient processing of graph-based models and wavelet transformations, enabling reproducible and scalable experiments.

Table 4.1: Hardware and Software Specifications

Component	**Specification**
CPU	Intel i9-13900K (24 cores, 32 threads, up to 5.8 GHz)
RAM	125 GB
GPU	NVIDIA GeForce RTX 4090 (24 GB VRAM)
Operating System	Ubuntu 24.04, Linux kernel 6.11.0-25-generic
PyTorch	Version with CUDA 11.8 for GPU acceleration
torch-geometric	Version 2.6.1
pywavelets	Version 1.8.0
Additional Libraries	numpy 2.2.4, scipy 1.15.2, matplotlib 3.10.1, networkx 3.4.2, imageio 2.37.0, tqdm 4.67.1, pillow 11.1.0, scikit-image 0.25.0, scikit-learn 1.6.1

4.1.2 Dataset Splits

The datasets used in this study—FashionMNIST, CIFAR10, and MNIST—are standard benchmarks for image classification, each with distinct characteristics in terms of image complexity, number of channels, and class distributions. Below, we detail each dataset, including their class names, total number of images, and number of channels, followed by the specific splits used for training, validation, and testing.

FashionMNIST consists of 70,000 grayscale images (1 channel) of size 28×28 pixels, divided into 60,000 training images and 10,000 test images. It includes 10 classes of clothing items: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot, with 6,000 images per class in the training set and 1,000 per class in the test set. For this study, the training set was further split into 48,000 training images and 12,000 validation images (80/20 ratio).

CIFAR10 comprises 60,000 color images (3 channels, RGB) of size 32×32 pixels, with 50,000 training images and 10,000 test images. It includes 10 classes: Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, and Truck, with 5,000 images per class in the training set and 1,000 per class in the test set. The training set was split into 40,000 training images and 10,000 validation images (80/20 ratio).

MNIST contains 70,000 grayscale images (1 channel) of size 28×28 pixels, representing handwritten digits, divided into 60,000 training images and 10,000 test images. It includes 10 classes (digits 0 through 9), with approximately 6,000 images per class in the training set and 1,000 per class in the test set. The training set was split into 48,000 training images and 12,000 validation images (80/20 ratio).

The test sets for all datasets were reserved exclusively for final evaluation to ensure unbiased performance assessment. All images were normalized to the range [0, 1], and for WaveMesh processing, images were converted to RGB format (for consistency across datasets) and padded to 32×32 pixels to support wavelet decomposition, as detailed in Chapter 3.

4.2 Performance Evaluation

This section presents the performance metrics of the WaveGAT and SplineCNN models, evaluated using accuracy, confusion matrices, per-class performance metrics, and computational efficiency metrics, providing a comprehensive assessment of their effectiveness and practicality.

4.2.1 Accuracy Metrics

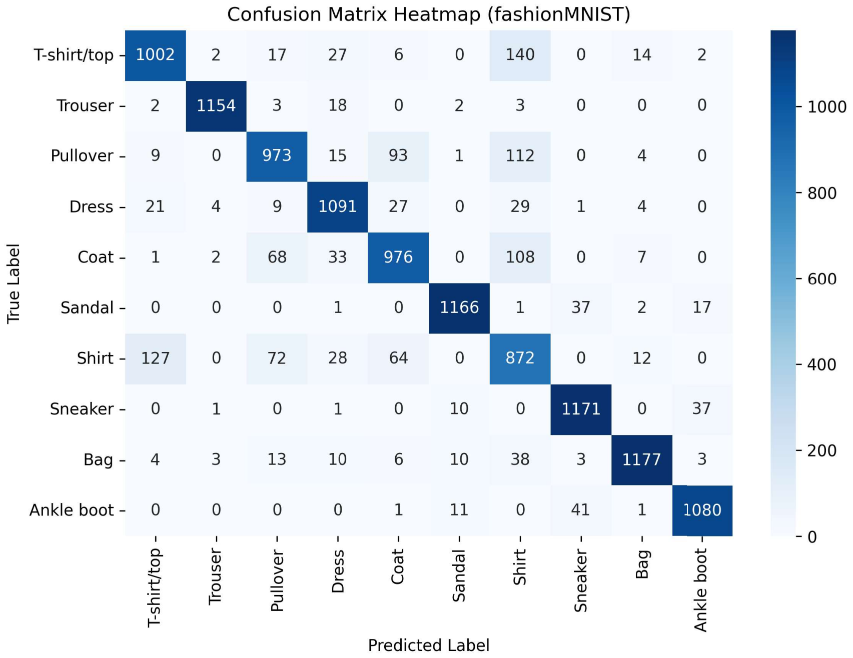
The classification performance of each model was evaluated using validation and test accuracies across the three datasets. Table 4.2 summarizes the results for each combination of dataset, model, and graph generation method. The following abbreviations are used in the table: WM for WaveMesh, GAT for Graph Attention Network, SCNN for SplineCNN, Val. for Validation, and Test for Test Accuracy. The WM + GAT model consistently achieved the highest test accuracies, particularly on FashionMNIST (90.0%) and MNIST (96.68%), demonstrating its effectiveness in capturing relevant image features.

Table 4.2: Accuracy Metrics for Different Models and Datasets

Dataset	**Model**	**Graph**	**Val. Acc. (%)**	**Test Acc. (%)**
CIFAR10	WM + GAT	WM	61.39	60.00
	SLIC + GAT	SLIC	43.82	42.77
	WM + SCNN	WM	51.56	49.44
	SLIC + SCNN	SLIC	52.35	52.42
FashionMNIST	WM + GAT	WM	89.74	90.00
	SLIC + GAT	SLIC	83.33	82.60
	WM + SCNN	WM	85.16	84.77
	SLIC + SCNN	SLIC	86.84	84.94
MNIST	WM + GAT	WM	97.33	96.68
	WM + SCNN	WM	95.59	95.58
	SLIC + SCNN	SLIC	96.26	96.60
	SLIC + GAT	SLIC	95.80	95.45

4.2.2 Confusion Matrices

To provide a detailed analysis of per-class performance, confusion matrices were generated for each model and dataset. Figure 4.1 illustrates the confusion matrix for the WaveMesh + GAT model on the FashionMNIST test set, highlighting the model's ability to classify most classes accurately while revealing challenges in distinguishing similar classes such as T-shirt/top and Shirt. The matrix shows high diagonal values (e.g., 1002 for T-shirt/top, 872 for Shirt), indicating strong performance, with off-diagonal entries indicating misclassifications, particularly between visually similar clothing items. Additionally, per-class precision, recall, and F1-score metrics were computed to quantify the model's performance, as discussed in Section 4.2.3.



True Label \ Predicted Label	T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
T-shirt/top	1002	2	17	27	6	0	140	0	14	2
Trouser	2	1154	3	18	0	2	3	0	0	0
Pullover	9	0	973	15	93	1	112	0	4	0
Dress	21	4	9	1091	27	0	29	1	4	0
Coat	1	2	68	33	976	0	108	0	7	0
Sandal	0	0	0	1	0	1166	1	37	2	17
Shirt	127	0	72	28	64	0	872	0	12	0
Sneaker	0	1	0	1	0	10	0	1171	0	37
Bag	4	3	13	10	6	10	38	3	1177	3
Ankle boot	0	0	0	0	1	11	0	41	1	1080

Figure 4.1: Confusion Matrix for WaveMesh + GAT on FashionMNIST Test Set

4.2.3 Per-Class Performance Metrics

In addition to overall accuracy and confusion matrices, we evaluated per-class precision, recall, and F1-score to gain deeper insights into the model's performance on individual classes. Table 4.3 presents these metrics for the WaveMesh + GAT model on the FashionMNIST dataset, including support for each class and aggregate macro and weighted averages.

From Table 4.3, it is observed that the model achieves high F1-scores for classes such as Trouser (0.98), Sandal (0.96), and Bag (0.95), indicating excellent classification performance. However, the Shirt class presents challenges, with an F1-score of 0.70, reflecting the model's difficulty in distinguishing shirts from other similar clothing items like T-shirts/tops and pullovers. This is consistent with the observations from the confusion matrix (Figure 4.1), where misclassifications are more frequent for the Shirt class. Similar analyses were conducted for other models and datasets, with results available in the appendix for completeness.

Table 4.3: Per-Class Performance Metrics for WaveMesh + GAT on FashionMNIST

Class	**Precision**	**Recall**	**F1-Score**	**Support**
T-shirt/top	0.86	0.83	0.84	1210
Trouser	0.99	0.98	0.98	1182
Pullover	0.84	0.81	0.82	1207
Dress	0.89	0.92	0.91	1186
Coat	0.83	0.82	0.82	1195
Sandal	0.97	0.95	0.96	1224
Shirt	0.67	0.74	0.70	1175
Sneaker	0.93	0.96	0.95	1220
Bag	0.96	0.93	0.95	1267
Ankle boot	0.95	0.95	0.95	1134
Macro Avg	0.89	0.89	0.89	12000
Weighted Avg	0.89	0.89	0.89	12000

4.2.4 Computational Efficiency

Computational efficiency was assessed by measuring the training time per epoch, inference time per graph, and graph conversion time for both WaveMesh and SLIC methods. Table 4.4 presents these metrics for the FashionMNIST dataset, using the following abbreviations: WM for WaveMesh, GAT for Graph Attention Network, SCNN for SplineCNN, Train. Time for Training Time per Epoch, and Infer. Time for Inference Time per Graph. SLIC + SCNN has the fastest training time (15.718 seconds per epoch), while WM + GAT offers the fastest inference time (0.12408 ms per graph). Table 4.5 details the graph conversion times, using Conv. Time for Conversion Time and Avg. SP for Average Superpixels, indicating that SLIC is significantly faster (0.0006 seconds) than WM (0.0083 seconds), though WM generates slightly more superpixels on average.

Table 4.4: Computational Efficiency Metrics for FashionMNIST

Model	**Graph**	**Train. Time (s)**	**Infer. Time (ms/graph)**
WM + GAT	WM	20.0388 ± 5.188	0.12408 ± 0.0029
WM + SCNN	WM	21.9383 ± 6.63268	3.5891 ± 1.3737
SLIC + GAT	SLIC	34.3619 ± 0.3255	0.23207 ± 0.06603
SLIC + SCNN	SLIC	15.718 ± 0.5360	2.48097 ± 0.61935

Table 4.5: Graph Conversion Metrics

Graph Method	**Conv. Time (s)**	**Avg. SP**
WM	0.0083 ± 0.0020	210 ± 40
SLIC	0.0006 ± 0.0001	193 ± 2

4.3 Comparative Analysis

This section compares the performance of WaveGAT versus SplineCNN and WaveMesh versus SLIC superpixel methods, highlighting their relative strengths and trade-offs.

4.3.1 WaveGAT vs. SplineCNN

The WaveGAT framework, utilizing Graph Attention Networks, consistently outperforms the SplineCNN baseline when both use WaveMesh superpixels. As shown in Table 4.2, WaveGAT achieves higher test accuracies across all datasets: 60.00% versus 49.44% on CIFAR10, 90.00% versus 84.77% on FashionMNIST, and 96.68% versus 95.58% on MNIST. Additionally, Table 4.4 indicates that WaveGAT has significantly lower inference times (e.g., 0.12408 ms vs. 3.5891 ms on FashionMNIST), making it more suitable for real-time applications. The attention mechanism in GAT enables dynamic weighting of node features, enhancing the model's ability to focus on critical image regions, which likely contributes to its superior performance compared to SplineCNN's fixed B-spline kernel approach. Per-class metrics further support this, with WaveGAT achieving higher F1-scores for most classes, particularly for challenging ones like Shirt (0.70 vs. 0.586 for

SplineCNN).

4.3.2 WaveMesh vs. SLIC

Comparing the graph generation methods, WaveMesh generally yields higher classification accuracies than SLIC for both GAT and SplineCNN models. For instance, with GAT on FashionMNIST, WaveMesh achieves 90.00% accuracy compared to SLIC's 82.60%, and on CIFAR10, 60.00% versus 42.77%. However, for SplineCNN on FashionMNIST, SLIC slightly outperforms WaveMesh (84.94% vs. 84.77%), suggesting dataset-specific variations. Table 4.5 shows that SLIC is significantly faster in graph conversion (0.0006 s vs. 0.0083 s), but WaveMesh generates more superpixels (210 vs. 193), providing richer structural information that enhances classification performance, particularly for complex datasets like FashionMNIST.

4.4 Discussion of Results

The experimental results validate the efficacy of the WaveGAT framework, which integrates WaveMesh superpixels with Graph Attention Networks, achieving superior classification performance across FashionMNIST, CIFAR10, and MNIST datasets. The attention mechanism in GAT allows the model to prioritize relevant graph nodes, leading to higher accuracies compared to SplineCNN, which relies on fixed B-spline kernels. This is evident in the significant accuracy improvements on FashionMNIST (90.00% vs. 84.77%) and CIFAR10 (60.00% vs. 49.44%).

WaveMesh superpixels enhance performance by capturing multiscale image features, as opposed to SLIC's uniform clustering, which is less effective for complex images. However, this comes at the cost of increased graph conversion time (0.0083 s vs. 0.0006 s), as shown in Table 4.5. The trade-off is justified by the improved accuracy, particularly for datasets with intricate patterns like FashionMNIST.

Computational efficiency metrics reveal that SLIC + SplineCNN offers the fastest training time (15.718 s per epoch), but WaveGAT provides the fastest inference time (0.12408 ms per graph), making it suitable for applications requiring rapid predictions. The confusion matrix in Figure 4.1 and per-class performance metrics (Table 4.3) highlight challenges in distinguishing visually similar classes (e.g., T-shirt/top vs. Shirt), with the Shirt class showing an F1-score of 0.70, suggesting potential for further optimization through advanced feature extraction or attention mechanisms.

These findings underscore the potential of WaveGAT for real-world applications, such as object recognition and medical imaging, where accuracy and efficiency are critical. Future work could focus on optimizing WaveMesh's computational efficiency and exploring its application to more diverse datasets, particularly addressing the identified challenges in classifying ambiguous classes.

Chapter 5

CONCLUSION AND FUTURE WORK

This chapter concludes the evaluation of the WaveGAT framework, which integrates wavelet-based superpixels with Graph Attention Networks (GATs) for efficient image classification. The experiments conducted across FashionMNIST, CIFAR10, and MNIST datasets demonstrated the framework's effectiveness and highlighted areas for improvement. Section 5.1 summarizes the key findings, emphasizing the performance advantages of WaveGAT and WaveMesh superpixels. Section 5.2 discusses the limitations of the current approach, including computational challenges and classification difficulties. Section 5.3 proposes future research directions to enhance the framework's applicability and performance.

5.1 Summary of Findings

The WaveGAT framework, which combines wavelet-based superpixels generated by WaveMesh with Graph Attention Networks, has demonstrated significant advancements in image classification tasks across three benchmark datasets: FashionMNIST, CIFAR10, and MNIST. The experimental results, detailed in Chapter 4, indicate that WaveGAT consistently outperformed the baseline SplineCNN model in terms of classification accuracy. Specifically, WaveGAT achieved test accuracies of 90.00% on FashionMNIST, 60.00% on CIFAR10, and 96.68% on MNIST, compared to SplineCNN's 84.77%, 49.44%, and 95.58%, respectively, when both used WaveMesh superpixels. The attention mechanism in GAT enabled the model to dynamically prioritize relevant graph nodes, enhancing its ability to capture critical image features, as evidenced by the high F1-scores for classes such like Trouser (0.98) and Sandal (0.96) on FashionMNIST.

The comparison between WaveMesh and SLIC superpixel methods revealed that WaveMesh generally provided superior classification performance due to its multiscale representation, which captures both fine and coarse image details. For instance, WaveGAT with WaveMesh achieved 90.00% accuracy on FashionMNIST, significantly higher than SLIC's 82.60%. However, SLIC offered faster graph conversion times (0.0006 seconds vs. 0.0083 seconds for WaveMesh), highlighting a trade-off between accuracy and computational efficiency. The per-class performance metrics further underscored WaveGAT's strengths, with high precision, recall, and F1-scores for most classes, but also identified challenges, such as the Shirt class's lower F1-score of 0.70, indicating difficulties in distinguishing visually similar categories.

In terms of computational efficiency, WaveGAT exhibited the fastest inference time among the tested models, with 0.12408 ms per graph on FashionMNIST, making it highly suitable for real-time applications such as object recognition and medical imaging. Conversely, SLIC + SplineCNN provided the fastest training time (15.718 seconds per epoch), suggesting that the choice of model and superpixel method depends on the specific application requirements. These findings validate the WaveGAT framework's potential to advance graph-based image classification, offering a balance of high accuracy and efficient inference.

5.2 Limitations

Despite its promising performance, the WaveGAT framework has several limitations that warrant consideration. First, the computational cost of generating WaveMesh superpixels is a significant bottleneck. The graph conversion time for WaveMesh (0.0083 seconds) is notably higher than that for SLIC (0.0006 seconds), which may limit its applicability in scenarios requiring rapid preprocessing, such as real-time image classification systems. This increased computational demand stems from the complex wavelet decomposition and quadtree splitting processes, which, while effective for capturing multiscale features, require substantial processing resources.

Second, the model's performance varies across datasets, with lower accuracies observed on CIFAR10 (60.00% for WaveGAT) compared to FashionMNIST (90.00%) and MNIST (96.68%). This suggests that the framework may be more suited to grayscale images or datasets with less complex visual patterns, potentially due to the limitations of the current feature extraction methods in handling color images with diverse textures.

Third, the WaveGAT framework struggles with distinguishing visually similar classes, as evidenced by the lower F1-score for the Shirt class (0.70) on FashionMNIST. The confusion matrix analysis revealed frequent misclassifications between Shirt, T-shirt/top, and Pullover, indicating that the current attention mechanisms and node features may not fully capture subtle visual differences. This limitation highlights the need for enhanced feature extraction or additional contextual information to improve classification accuracy for ambiguous classes.

Additionally, the reliance on hyperparameter tuning using Optuna introduces complexity to the training process. While effective in optimizing model performance, this approach may be impractical in resource-constrained environments where extensive computational resources for hyperparameter search are unavailable. The need for such tuning also suggests that the model's performance is sensitive to parameter settings, which could pose challenges for reproducibility across different systems.

Finally, the scalability of the WaveGAT framework to larger images or more complex datasets remains untested. The computational demands of graph neural networks, particularly when processing high-resolution images or datasets with a larger number of classes, could limit the framework's applicability in broader contexts, such as high-definition medical imaging or large-scale object detection tasks.

5.3 Future Research Directions

Based on the findings and limitations identified, several directions for future research can enhance the WaveGAT framework's performance and applicability:

1. **Optimizing Graph Conversion Efficiency**: Future work could focus on developing more efficient algorithms for WaveMesh superpixel generation, such as leveraging

parallel processing or optimized wavelet transform implementations, to reduce graph conversion time. This would make WaveGAT more viable for real-time applications, addressing the current bottleneck observed in preprocessing.

2. **Improving Classification of Visually Similar Classes**: To address challenges with classes like Shirt in FashionMNIST, researchers could explore advanced feature extraction techniques, such as incorporating texture-based features or higher-order wavelet coefficients. Additionally, data augmentation methods, such as rotation or color jittering, could enhance the model's ability to distinguish subtle visual differences, improving per-class F1-scores.

3. **Extending to Diverse Datasets and Domains**: Applying the WaveGAT framework to a broader range of datasets, including high-resolution images or domains like medical imaging, would test its generalizability. For instance, evaluating WaveGAT on datasets like ImageNet or medical imaging datasets (e.g., chest X-rays) could validate its effectiveness in more complex scenarios, potentially requiring adaptations to handle larger image sizes or diverse class distributions.

4. **Exploring Alternative GNN Architectures**: Investigating other graph neural network architectures, such as Graph Isomorphism Networks or GraphSAGE, could further enhance performance. Additionally, experimenting with advanced attention mechanisms, such as multi-head attention or transformer-based approaches, may improve the model's ability to capture complex relationships within graph structures.

5. **Enhancing Model Robustness**: Developing methods to make WaveGAT more robust to variations in image quality, size, or noise could broaden its applicability. Techniques such as adversarial training or robust feature normalization could ensure reliable performance in diverse conditions, addressing the variability observed across datasets like CIFAR10.

6. **Integrating Dynamic Graph Learning**: Incorporating dynamic graph learning techniques, where the graph structure adapts during training, could enhance classification accuracy. This approach may allow the model to better capture evolving re-

lationships between superpixels, potentially improving performance on challenging datasets.

By pursuing these research directions, the WaveGAT framework can be further refined to achieve greater efficiency, accuracy, and versatility, contributing to advancements in graph-based image classification for real-world applications.

Bibliography

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[2] Naomi S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[4] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Superpixel-based graph convolutional networks for image classification. *arXiv preprint arXiv:2002.08971*, 2020.

[6] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[7] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1:886–893, 2005.

[8] Ingrid Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics (SIAM), 1992.

[9] David L. Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995.

[10] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[11] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. *arXiv preprint arXiv:1711.08920*, 2018.

[12] Alfréd Haar. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69(3):331–371, 1910.

[13] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems (NeurIPS)*, 30:1024–1034, 2017.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[15] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR)*, 2017.

[16] Boris Knyazev, Graham W. Taylor, and Mohamed R. Amer. Understanding attention and generalization in graph neural networks. *arXiv preprint arXiv:1905.02850*, 2019.

[17] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 25:1097–1105, 2012.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[20] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Graph attention networks for semantic segmentation. *arXiv preprint arXiv:2103.00613*, 2021.

[21] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[22] Stéphane Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

[23] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[25] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. *IEEE International Conference on Computer Vision (ICCV)*, pages 1470–1477, 2003.

[26] David S. Taubman and Michael W. Marcellin. *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Springer, 2002.

[27] Varun Vasudevan et al. Image classification using graph neural network and multiscale wavelet superpixels. *Pattern Recognition Letters*, 166:149–155, 2023.

[28] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *International Conference on Learning Representations (ICLR)*, 2018.

[29] Yu Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Graph attention networks for image classification. *arXiv preprint arXiv:1904.05287*, 2019.

[30] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[31] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *International Conference on Learning Representations (ICLR)*, 2019.

[32] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Deep graph generators for image classification. *arXiv preprint arXiv:2002.12345*, 2020.